① Machine learning foundations - Washington week 1

Getting started with IPython Notebook GraphLab and SFrame

SHIFT + Enter = executes the command

Enter: creates a multiline code (doesn't execute)

Cell + New Cell + Markdown: write code in markdown

- Python is a scripting language, easy to prototype things
  - or **ESC (ECHAP) + M**

None type variable: n = None

Advanced printing: print "Our float value is %.s. Our int value is %.s." % (f, i) → variables name

else if = elif

counter += 1: counter = counter + 1

Lambda function (other way to define a function):

E.g. square = **lambda x**: x * x . square(3) = 9

SFrames: scalable data structures for dealing with large datasets. It goes directly in the disk, you don't need lots of memory. It is part of GraphLab Create package.

dataframe: dataframe.head() = print the first few lines

dataframe.tail() print the last few lines.

② Machine Learning Foundations - Washington

week 1
week 2

GraphLab Canvas

- Canvas for data visualization. It can take any data structure.

dataset.show() = open a new web page that represents the data visually with some interesting info = histogram, mean, most common value... You can play with the canvas

To show this canvas directly inside the notebook

```
graphlab.canvas.set_target('ipynb')
```

to mean python notebook

With that, you cannot see all but select a category inside the canvas. E.g. dataset['column'].show(view="Categorical") one subgraphic view in the canvas

or dataset['column'].show()

Apply function: dataset['column'] = dataset['column'].apply(function)

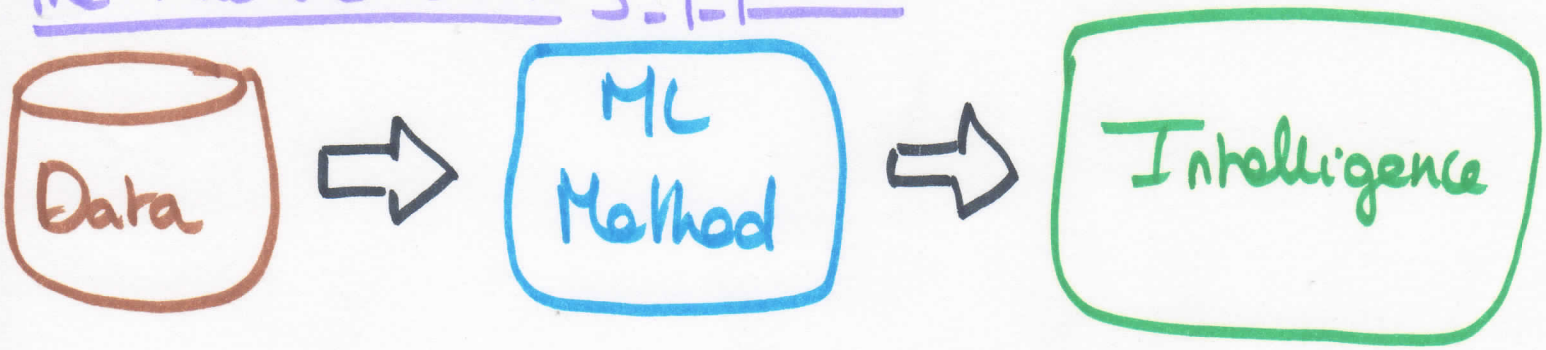
Regression

We have some set of features and we want to model how an observation that are associated with those features change as we change the values of the features. feature = covariate = predictor

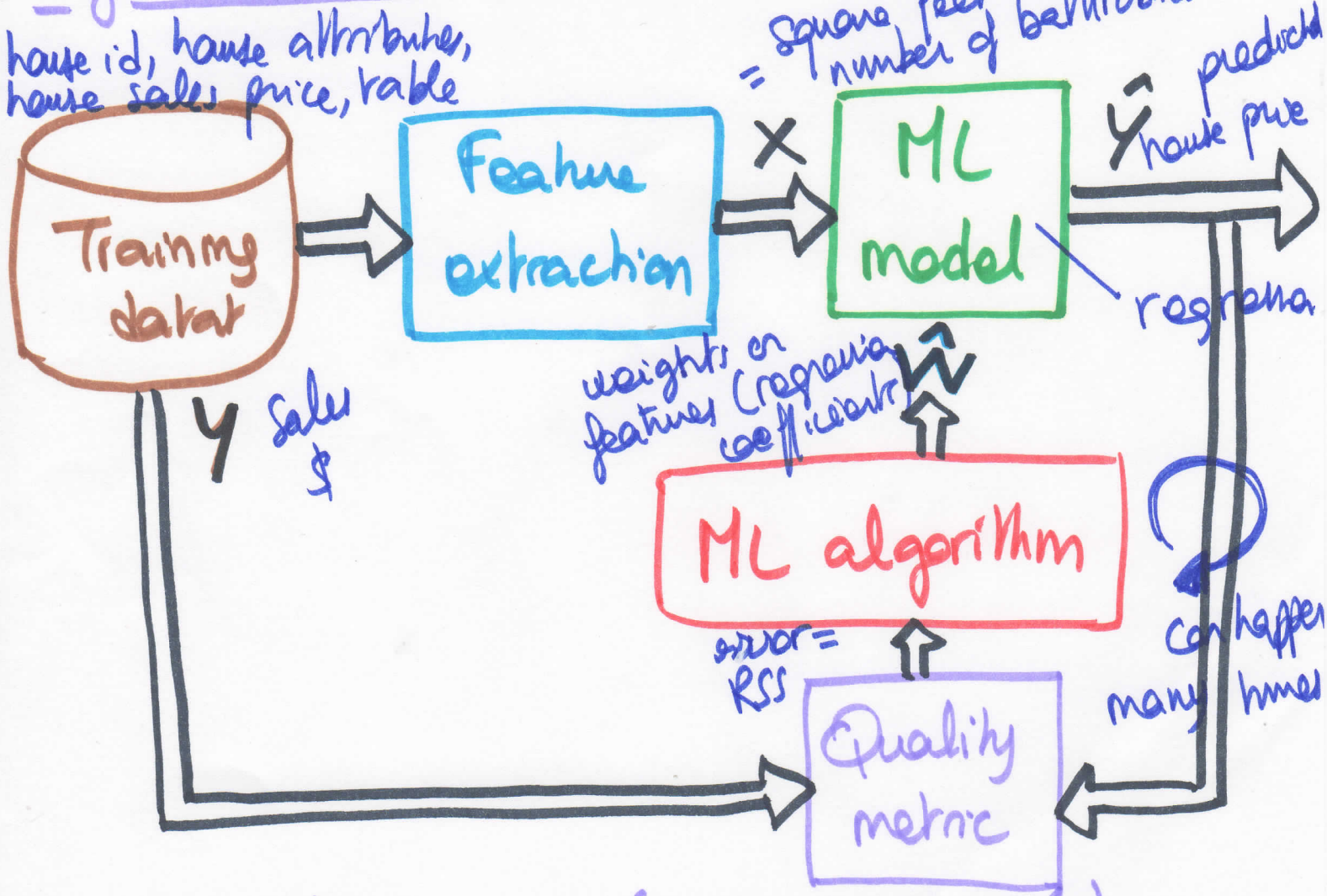
③ Machine learning foundations - Washington

Week 1

The Machine Learning pipeline



Regression Machine Learning diagram



GraphLab house pricing (regression example)

To get some visualization: dataset.show(view="Scatter Plot", x="column x", y="column y") → don't forget graphlab.canvas.set_target("ipynb")

④ Machine learning_Foundations - Washington

week 2

- You can get more details sliding the mouse on the data points and save the visualization as an image.

Splitting the data into training and test sets

```
train_data, test_data = dataset.random_split(
.8, seed = 2015)
```

↑ for 80% / 20%

→ random number

Build a simple regression model

```
model = graphlab.linear_regression.create(train_data,
target = dependantvariable, features = ['independantvar'])
```

Evaluate the model (on the test data)

```
model.evaluate(test_data)
{'max_error': ....., 'rmse': ....}
```

↑ outlier

Visualizing predictions with Matplotlib

- `import matplotlib.pyplot as plt`

  piece of the package

- `%matplotlib inline`

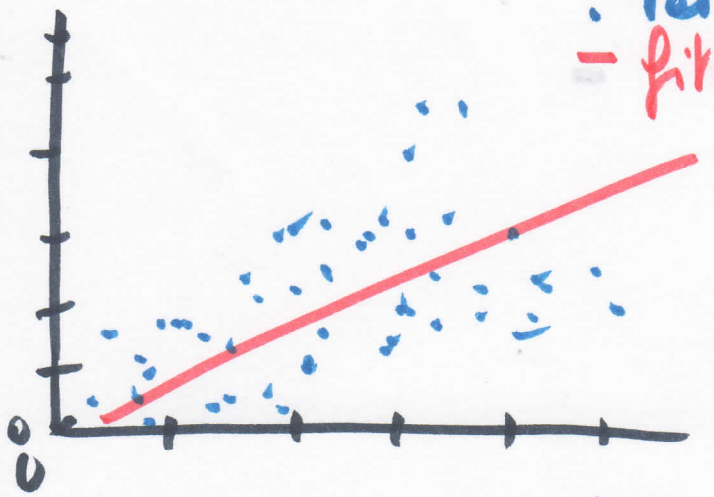
  plot directly in the notebook

- `plt.plot(test_data['independantvar'], test_data['dependantvar'], '.', test_data['independantvar'], model.predict(test_data), '-')`

  dot like scatterplot

  filled regression line

⑤ Machine learning_foundations - Working to week 2



Get the coefficients of the model

```
model.get("coefficients")
```

Exploring the features of the data

```
my_features = ["feature 1", "feature 2", ... ]
dataset[my_features].show()
```

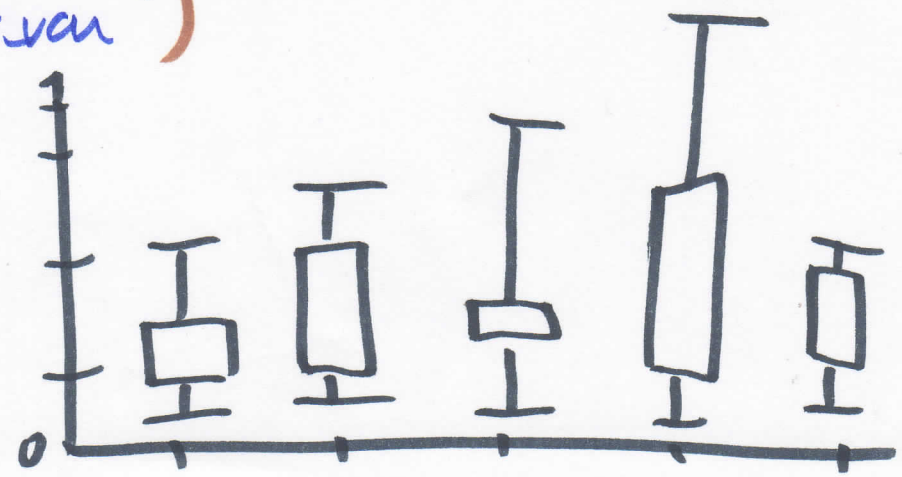
↳ you will get a nb of info on each feature (min, max, min, frequent items ...)

Boxplots

```
dataset.show(view="Box Whisker Plot", x="feature", y="dependant var")
```

↑ or independant var

- You can also play with the plot if lots of values



⑥ Machine Learning - Foundation - Washington

week 2/3

Build a linear regression model with multiple features

```
my_features_model = graphlab.linear_regression.
create(train_data, target = 'price', features =
my_features)
```

↳ from previously

HTML in IPython notebook

You can imbed HTML, images ... in the notebook

```
<img src = "image.jpeg">
```

(e.g) → markup

⚠ type ECHAP + M for the code to work

Classification

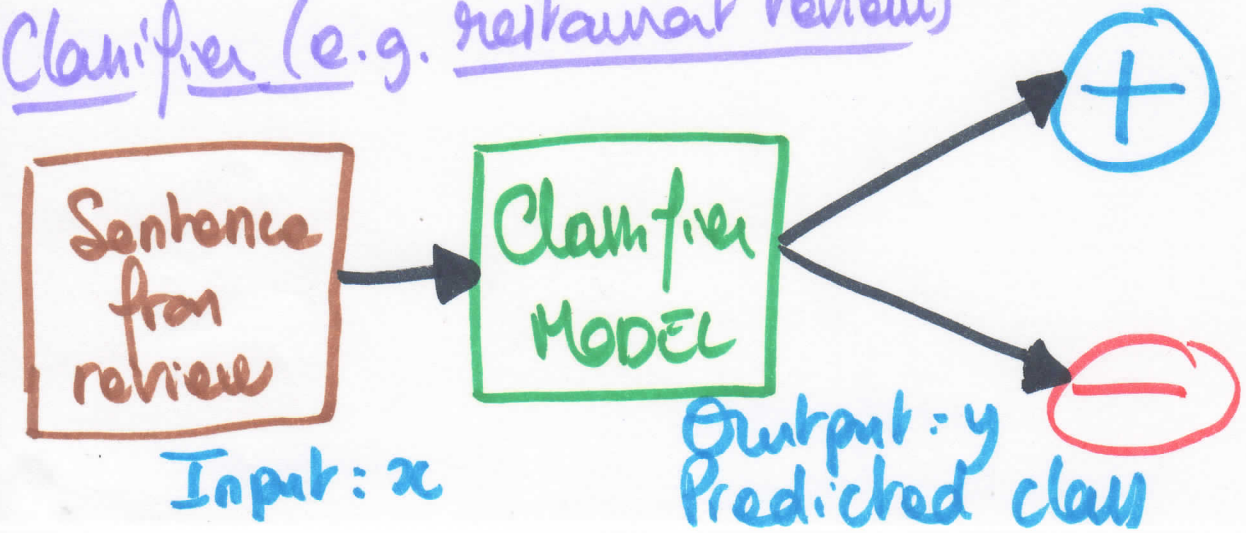
One of most common/used areas of Machine Learning.

Intelligent restaurant review system

All reviews for restaurant ⇨ Break all reviews into sentences ⇨ Sentence Sentiment classifier ⇨ Average predictions

Most ⊕ and ⊖

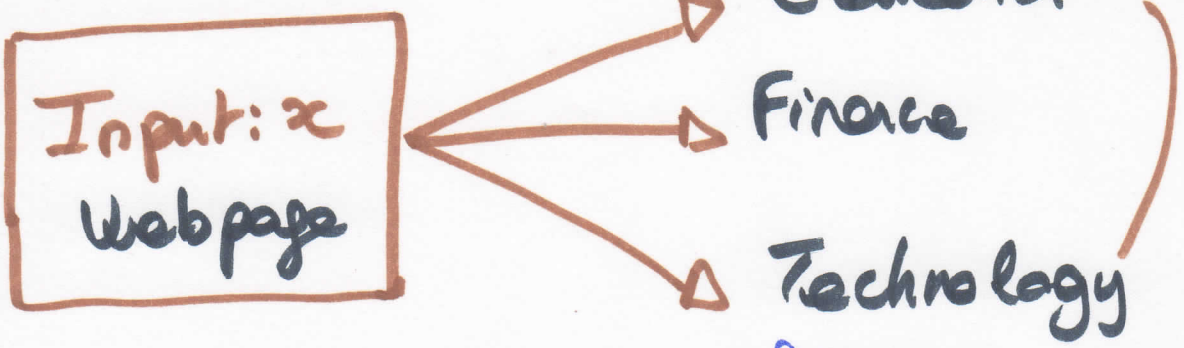
Classifier (e.g. restaurant review)



⑦ Machine learning foundations - Washington week 3

Multiclass classifier

The output y has more than 2 categories



- **Classifier examples**: spam filtering, image classification (through image pixels that recognize what it is), medical diagnosis (disease classifier)

Simple threshold classifier - restaurant review ex

List of positive words	List of negative words
great, awesome, good, amazing...	bad, terrible, disgusting, sucks...

Sushi was great, the food was awesome, but the service was terrible

Simple threshold classifier

Count positive & negative words in sentence

If number of positive words > number of negative words $\hat{y} = +$

Else $\hat{y} = -$

⑧ Machine learning - foundations - Washington week 3

Problems with threshold classifier

- How do we get a list of positive/negative words?
- Words have different degrees of sentiment:
  - Great > good
  - How do we weight different words?

Addressed by learning a classifier

- Simple words are not enough
  - Good ⇒ Positive
  - Not good ⇒ Negative

Addressed by more elaborate features

Simple linear classifier

Word	Weight
good	1.0
great	1.2
⋮	⋮

- It is called a linear classifier, because the output is weighted sum of input.

Sentence from review → Input: x

Simple linear classifier

Score(x) = weighted count of words in sentence

if Score(x) > 0 : $\hat{y}$ = ⊕

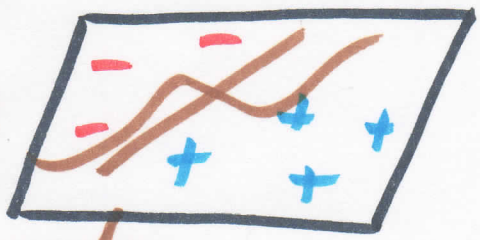
Else : $\hat{y}$ = ⊖

⑨ Machine learning foundations - Washington

week 3

Decision boundary separates positive & negative predictions

- For linear classifiers: 3 dimensions
  - When 2 weights are non-zero: line
  - When 3 weights are non-zero: plane
  - When many weights are non-zero: hyperplane



- For more general classifiers
  - More complicated shapes

Classification error and accuracy (reviews example)

- Error measures fraction of mistakes:
  - Best possible value is 0.0

$$\text{error} = \frac{\#\text{ of mistakes}}{\text{Total \# of sentences}}$$

- Measure accuracy - fraction of correct predictions
  - Best possible value is 1.0

$$\text{accuracy} = \frac{\#\text{ of correct}}{\text{Total \# of sentences}}$$

What if you just guess?

- For binary classification: accuracy = 0.5
- For k classes, accuracy = 1/k

What is a good accuracy?

A classifier with 90% accuracy is not always good. E.g. in spam email (to check)

① Machine learning foundations - Washington week 3

- Always be digging in and asking the hard questions about reported accuracies.
  - Is there a class imbalance?
  - How does it compare to a simple baseline approach? (random guessing, majority class...)
  - Most importantly = what accuracy does my application need? (What is good enough for my user's experience? What is the impact of the mistakes we make?..)

Confusion matrix - False positives, false negatives

- Cost of different types of mistakes can be different (& high) in some applications.

	Spam filtering	Medical diagnosis
False negative	Annoying	Disease not treated
False positive	Email lost Higher cost	Wasteful treatment

True label \ Predicted label	+	−
+	TP	FN
−	FP	TN

- Multiclass classification

100 test examples

True label \ Predicted label	Healthy	Cold	Flu
Healthy 70	60	8	2
Cold 20	4	12	4
Flu 10	0	2	8

accuracy = 60 + 12 + 8 = 80 / 100

(11) Machine learning foundations - Washington

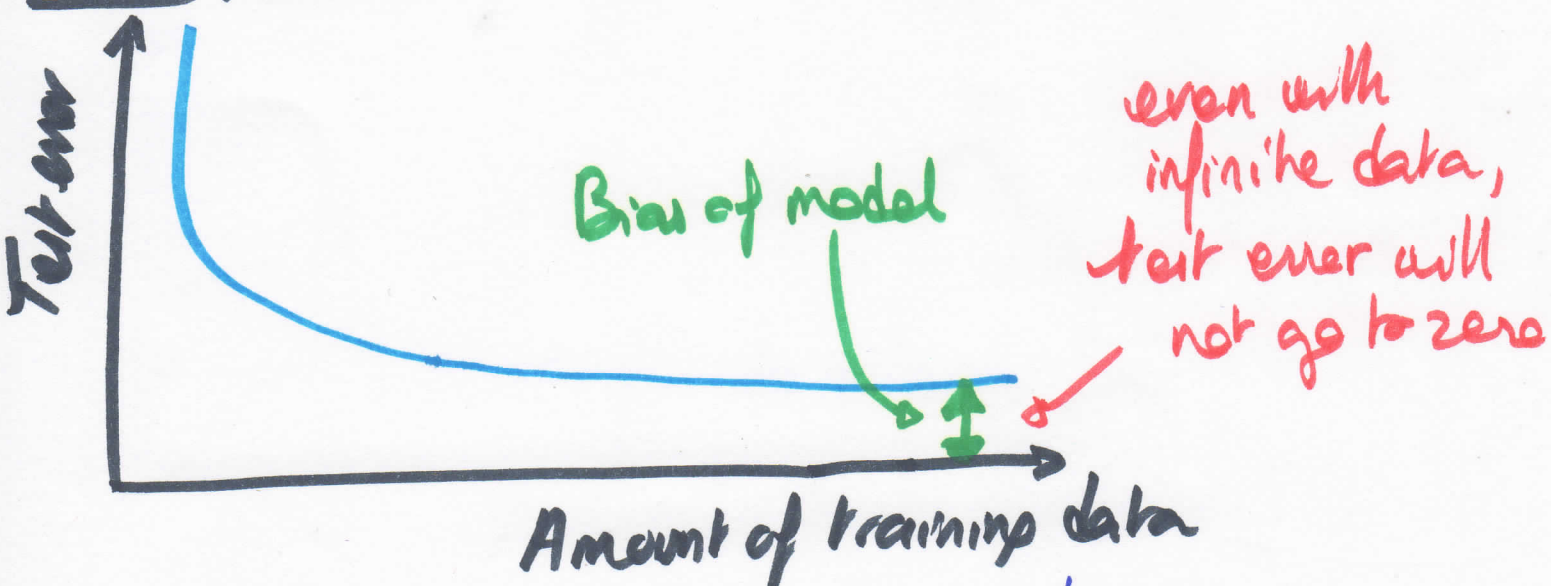
Week 3

Learning curves: How much data do I need?

How much data does a model need to learn?

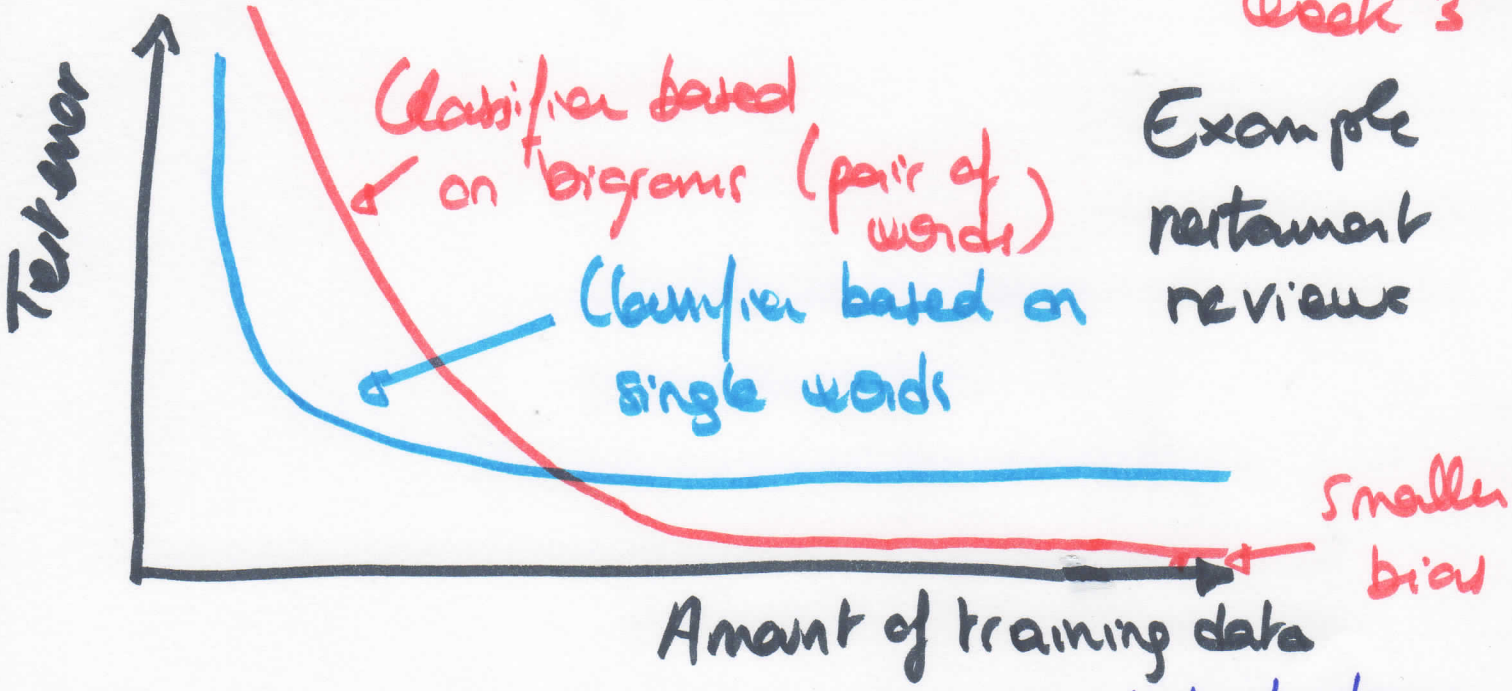
- **The more the merrier**
  - But data quality is the most important factor ⚠
    Better less data but with more quality
- **Theoretical techniques sometimes can bound how much data is needed**
  - Typically too loose for practical application
  - But provide guidance
- **In practice**
  - More complex models require more data
  - Empirical analysis can provide guidance

Learning curve



- More complex models tend to have less bias

(12) Machine learning foundations - Washington Week 3



- Models with less bias tend to need more data to learn well, but do better with sufficient data.

Class probabilities

- How confident is you prediction? → + or −
- Many classifiers provide a confidence level:

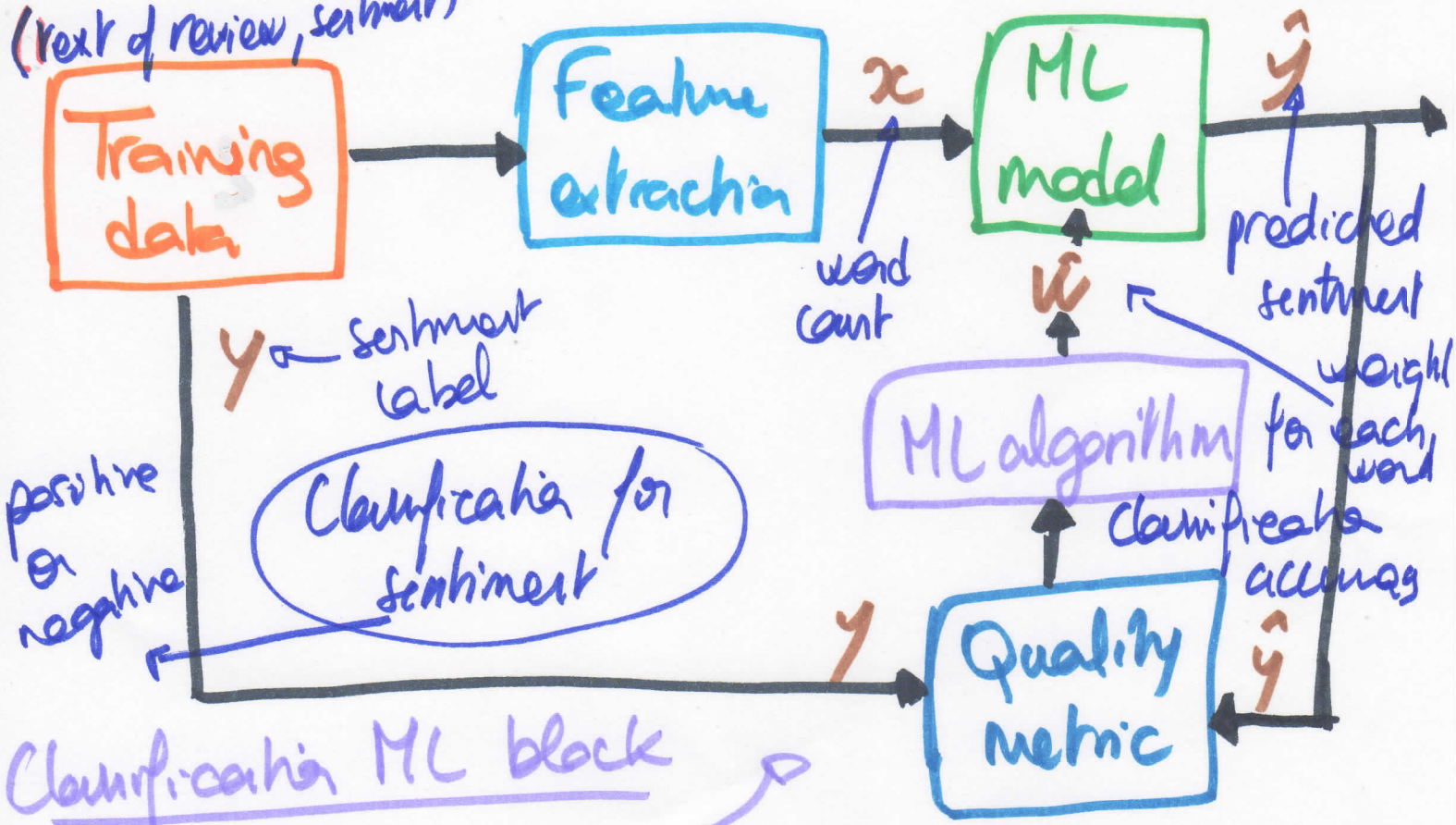
$$P(y = + \text{ or } - \mid x) = ? \quad 0:1$$

output label — Input sentence

E.g: $P(y = + \mid x) = 0.99$ High confidence

- It is extremely useful in practice.

(13) Machine learning foundations - Washington week 3



Classification ML block diagram

Build a word count vector

the column you want to analyze. ★

graphlab.text_analytics.count_words(dataset["column"])

- It will return a list with pairs of word / number of times this word appears. (You can create a new column with this data - word count vector)

Build a classifier in graphlab python

classifier_model = graphlab.logistic_classifier.create(train_data, target="column", features="column", validation_set=test_data)

Add [] if the feature type is not a list (to convert it to a list)

(14) Machine learning foundations - Washington

week 3/4

Roc curve = evaluate of model

- sentiment_model.evaluate(test_data, metric = "roc_curve")
- get some data, but for visualization with ability to change the threshold + get the confusion matrix:
- sentiment_model.show("Evaluation")

Get the probabilities of the prediction

model.predict(dataset, output_type = "probability")

Get the last row / second last row... in a dataframe

dataset[-1] dataset[-2] ...

get last row get second last row

Document retrieval

Currently reading an article you like.
Goal: you want to find similar article
Challenges: How do we measure similarity?
How do we search over articles?

Word count representation for measuring similarity

- Bag of words model: ignore order of words
- count # of instances of each word in vocabulary

0	1	2	0	0	2	2	0	1	0	0	1	...
carlos	the	free	calls	sport	cat	futbol				soccer		

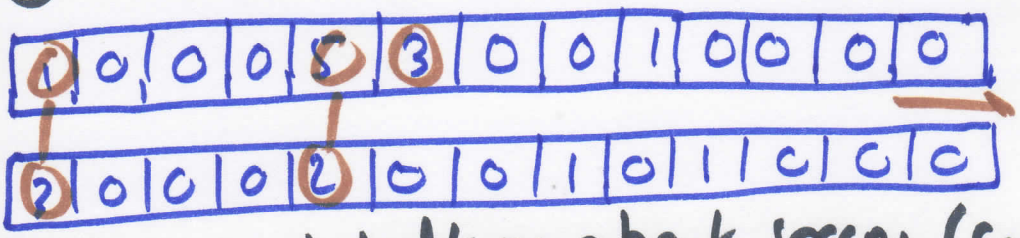
"Carlos calls the sport futbol. Emily calls the sport soccer."

⑤ Machine learning foundations - Washington week 4

Measuring similarity:

① Document talking about soccer

1)



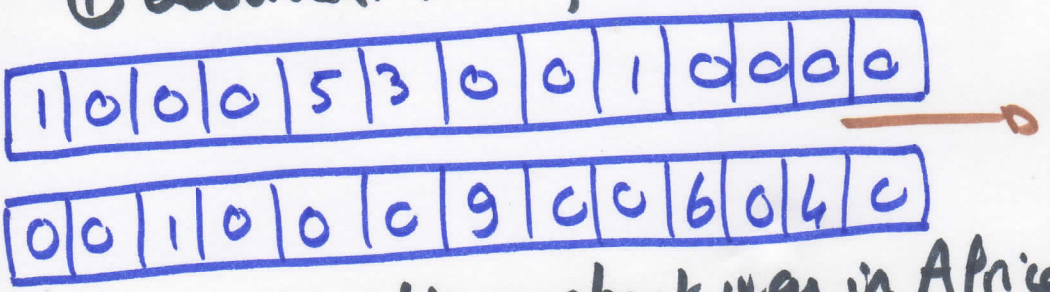
$1 \times 3 + 5 \times 2$

$= 13$ similarities

② Document talking about soccer (some players)

2)

① Document talking about soccer



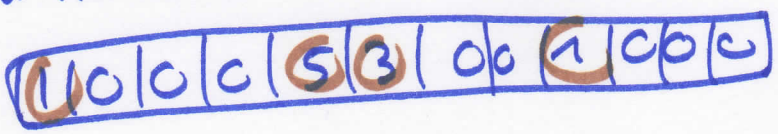
$= 0$ no similarity

② Document talking about war in Africa

Issues with word counts: document length

The more the document is large the more the "similarity" (eagles" will be high (as numbers will be higher for words)

Solution: normalize



$$\sqrt{(1^2 + 5^2 + 3^2 + 1^2)}$$

Prioritizing important words with tf-idf

Issues with word counts: Rare words

Common words in doc: "the", "player", "field".. dominate

rare words like: "futbol", "Messi"

(16) Machine Learning Foundations - Machine Learning Week 4

Document frequency:

- What characterize a rare word?
  - appear infrequently in the corpus
- Emphasize word appearing in few docs
  - equivalently, discount word w based on # of docs containing w in corpus

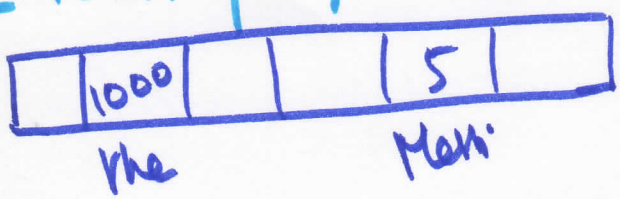
Important words:

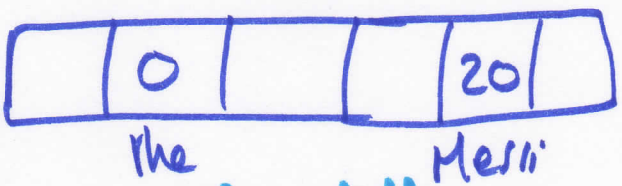
- Do we want only rare words to dominate?
- What characterize an important word?
  - Appear frequently in document (common locally)
  - Appear rarely in corpus (rare globally)
- Trade off between local frequency and global rarity

TF-IDF document representation

- Term Frequency - Inverse Document Frequency (tf-idf)

- Term frequency

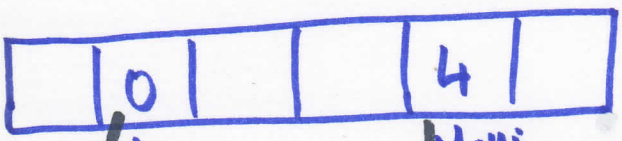




tf x idf

- Inverse document frequency

64 documents



$\log\left(\frac{64}{1+63}\right)=0$ $\log\left(\frac{64}{1+3}\right)\cdot\log(16)$

$$\log\left(\frac{\#\text{docs}}{1+\#\text{docs using the word}}\right)$$

(17) Machine learning foundation - Washington week 4

Idf log
- word in many docs $\log\left(\frac{\text{large \#}}{1+\text{large \#}}\right) = \log 1 = 0$
- rare word $\log\left(\frac{\text{large \#}}{1+\text{small \#}}\right) = \text{large \#}$

Retrieving similar documents

1- Nearest neighbor:

Input: Query article
Output: Most similar article (among the corpus)
Algorithm: Search over each article in corpus
- Compute s = similarity (query article, article in corpus)
- if $s >$ Best_s, record Most similar article = article in corpus and set Best_s = s
- Return Most similar article

K- Nearest neighbor

Input: Query article
Output: List of K similar articles

Nearest neighbor search: query article - corpus
specify: distance metric
Output: set of most similar articles

(18) Machine learning foundations - Clustering week 4

Structure docs by topics: supervised learning

Structure documents by topic. Discover groups (clusters) of related articles. ≠ class

Training set of labeled docs: sport, world news, science ..

- It is a multiclass classification problem

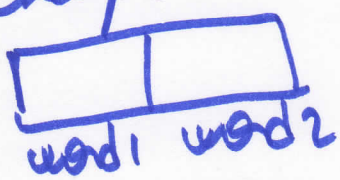
Clustering documents: unsupervised learning

- No labels provided
- Want to uncover cluster structure

Input: docs as vectors

Output: cluster labels

Example with 2-dimension vector



What defines a cluster?

- Cluster defined by center & shape/spread
- Assign observation (doc) to cluster (topic label)
  - Score under cluster is higher than other
  - Often, just more similar to assigned cluster center than other cluster centers

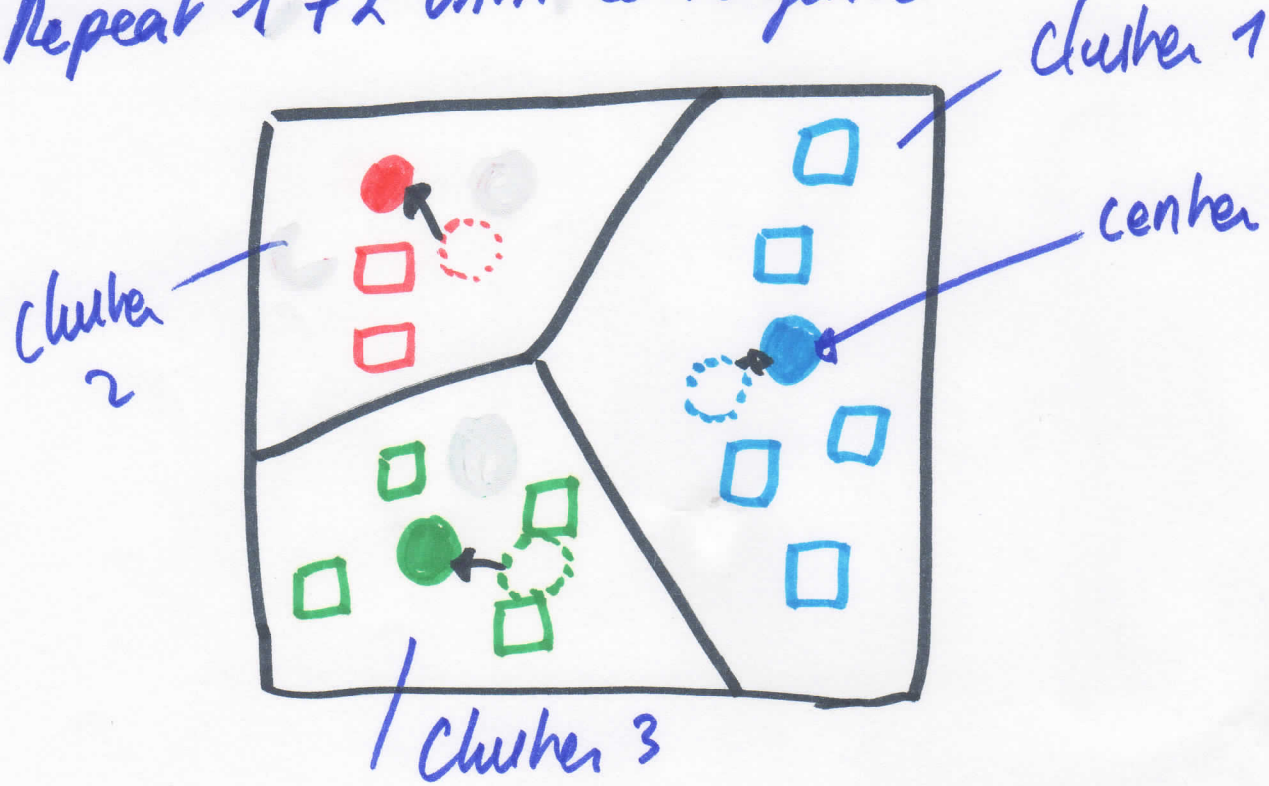
K-means algorithm

- Assume - similarity metric = distance to cluster center (smaller is better)

(19) Machine learning foundations - Washington week 4

K-means algorithm

0. Initialize cluster centers
1. Assign observations to closest cluster center
2. Revise cluster centers as mean of assigned observations
3. Repeat 1 + 2 until convergence

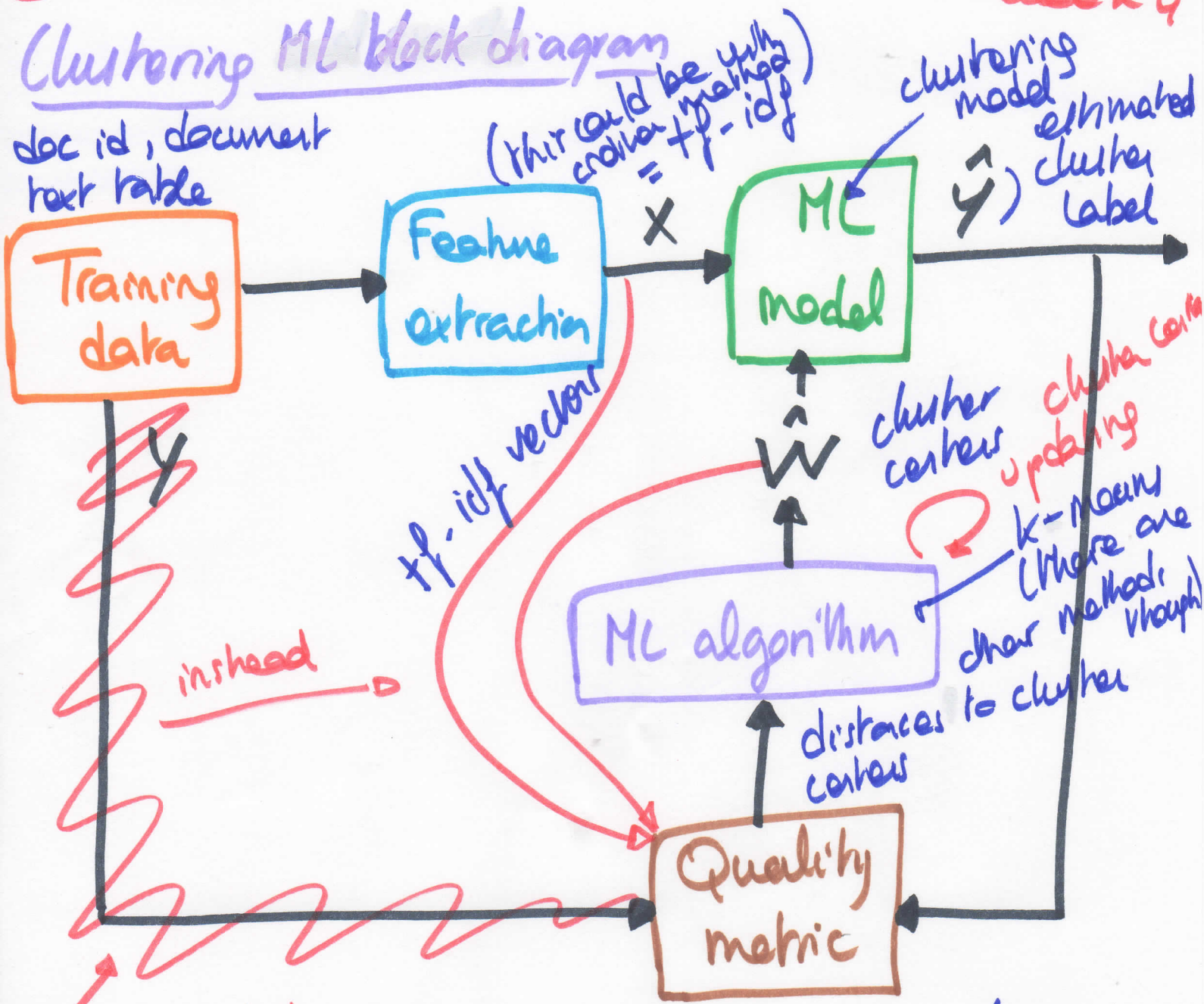


Other clustering examples

- Clustering images, for search, group as: ocean, dog...
- Grouping patients by medical condition (to better characterize subpopulation and disease)
- Products on Amazon (discover product categories from purchase histories, discovering groups of users)
- Structuring web search results (search terms can have multiple meanings).

② Machine learning foundations - Washington week 4

Clustering ML block diagram



Graphlab SFrame print ALL rows and columns

```
SFrame.print_rows(num_rows=m, num_columns=n)
```

(21) Machine learning_foundations - Washington week 4

Python graphlab - Divide a column in two

Dictionary made of tuples → column 1 = key. column 2 = ?

dataset[['column name']].stack('column name', new_column_name = ['column 1', 'column 2'])

Python graphlab: Compute TF-IDF

Have the column 'word_count' ready

tfidf = graphlab.text_analytics.tf_idf(dataset['word_count'])

It returns a dictionary: You can then use .stack to divide it by tfidf into 2 columns: "word" and "tfidf", and sort tfidf into 2 columns: "word" and "tfidf", and sort it by tfidf

Python graphlab: Compute distance TF-IDF

Manually, between two tfidf

graphlab.distances.cosine(dataset1['tfidf'][0], dataset2['tfidf'][0])

to transform it into a dictionary (instead of an SArray)

cosine distance = the lower the cosine distance, the closer the articles are. It measures similarity between vectors

. There are other techniques.

(22) Machine Learning Foundations – Washington week 4 5

Python graphlab · build a k nearest neighbor model

```
knn_model = graphlab.nearest_neighbors.create(
dataset, features = ['tfidf'], label = 'name')
```

- tfidf: column in the dataset with tfidf values
- label: good for the example in the lecture, might be ≠

Query an input to find nearest

```
knn_model.query(input)
```

The input in the lecture video is the name of the person

E.g. `arnold = people[people['name'] == 'Arnold Schawarzenegger']`
(arnold = input)

Recommender systems : use cases examples

- Personalization is transforming our experience of the world
- Youtube - videos recommendation.

100 hours of videos are uploaded every minute.
What do I care about? There is information overload.
Browsing is "history": viewers | videos

Personalization: connects users & items

- Movie recommendations - Netflix

Connect users with movies they may want to watch

- Product recommendations - Amazon (the first to have ≃ build a recommender system)

(23) Machine learning foundations - Washington weeks

Recommendations combine global & social interests

Music recommendations - Pandora

Recommendations from coherent & diverse sequence

Friend recommendations - Facebook

Users and "items" are of the same "type":
Build a network of users.

Drug-target interactions

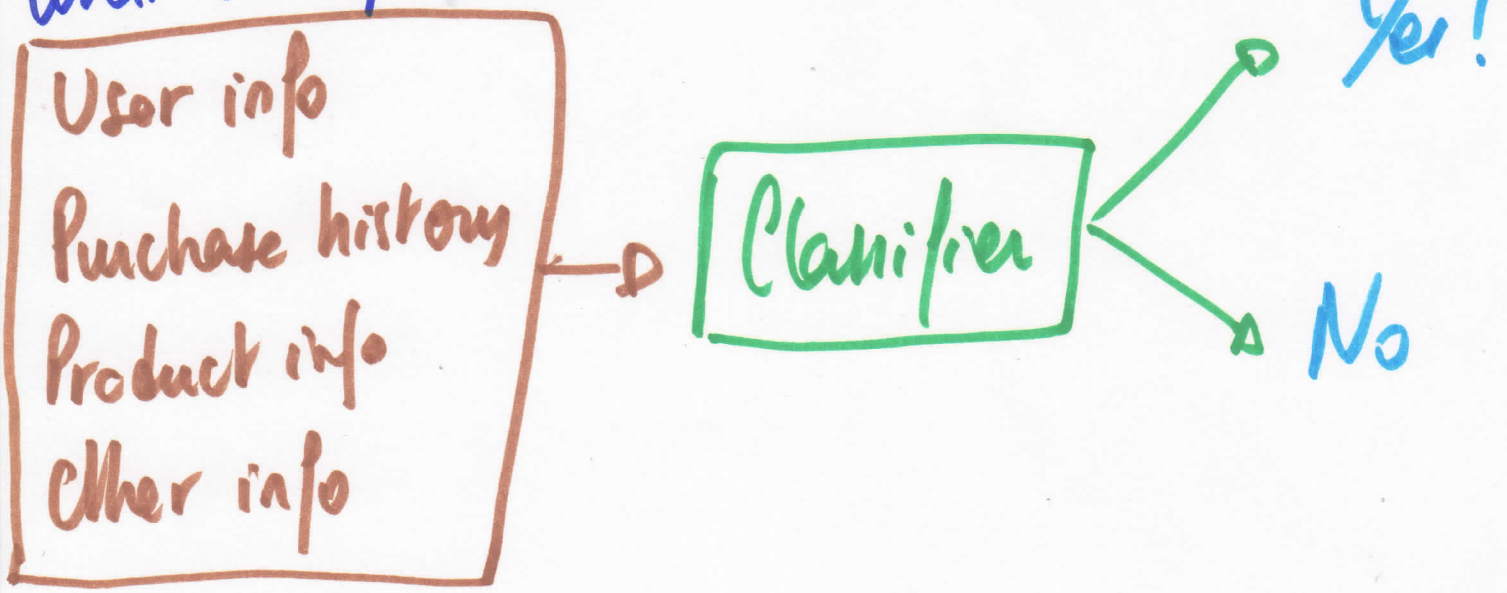
What drug should we "repurpose" for some disease?

Recommender system - Popularity (solution 0)

- It is the simplest approach. Used by News websites.
- What are people viewing now? Rank by global popularity. Limitations: no personalization "Most popular articles"

Recommender system - Classification model (sol. 1)

What is the probability I'll buy this product?



(24) Machine learning foundations - walkthrough weeks

Pros: . Personalized (Consider user info & purchase history)
. Features can capture context: Time of the day what I just saw... . Even handles limited user history: Ages of user...

Limitations: . Features may not be available
. Often doesn't perform as well as collaborative filtering methods (next)

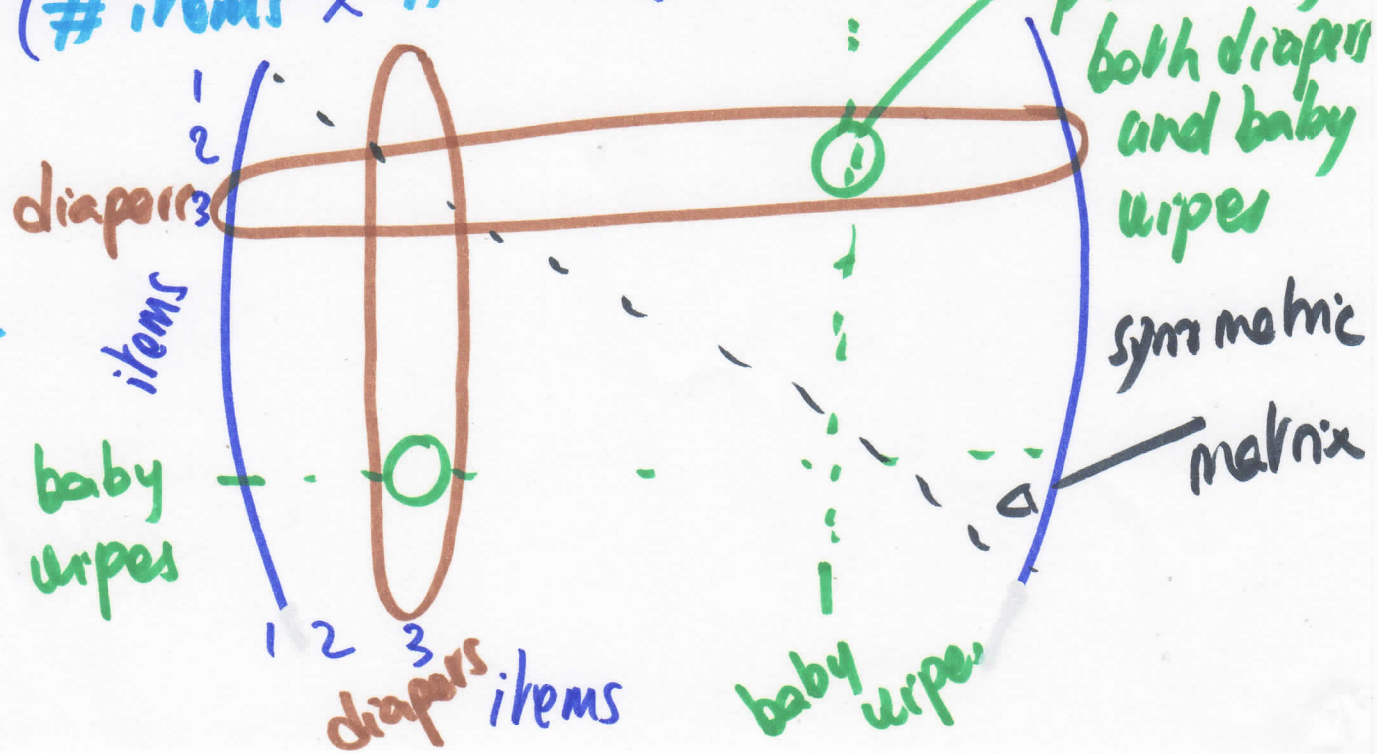
Recommender system = collaborative filtering (sol. 2)

People who bought this, also bought....

Co-occurrence matrix - example

. People who bought diapers also bought baby wipes

Matric C: store # users who bought both items
i & j - (# items x # items) matrix



(25) Machine learning Foundation - Washington Week 5

Matrix C is symmetric: # purchasing i & j same as # for j & i ($C_{ij} = C_{ji}$)

Building their co-occurence matrix: search for all history of users of purchase they've made and count. In this example, every time we see a purchase of diaper we'll add it to the matrix, and keep incrementing the matrix this way as we are search through user history of purchase.

Making recommendation using co-occurences

- User purchased diapers.

① look at diapers row of matrix ← people who bought diapers also bought all of these products

[.. 0 .. ...4 ... 100 ...]

DVD pacifiers baby wipes

② Recommend other items with largest counts (sorting the row) - here it could be baby wipes, milk.
↳ for somebody who just purchased a product.

Co-occurence matrix must be normalized

- What if there are very popular items?

Example: - Popular baby item: diapers
- For any baby item i = Sophie giraffe large count C_{ij} for j = diapers

(26) Machine learning foundations - Walking to week 5

Sophie giraffe [0 .... Imillia .. ..]
DVD diapers baby wipes

The recommender system will then be very likely to recommend diapers.

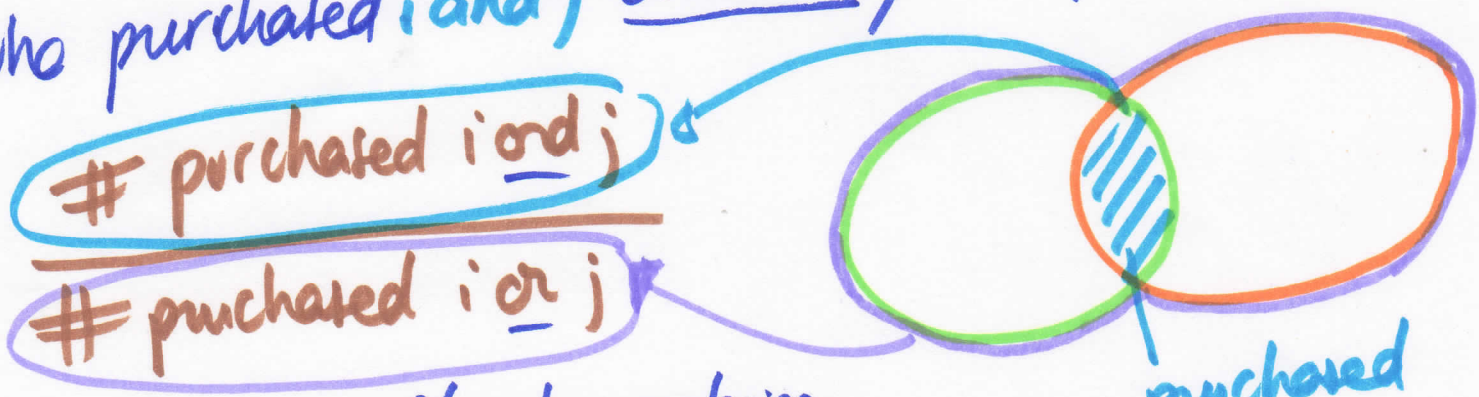
Results:

- Having those very large counts for popular items drowns out all the other effects.
- We end up with a recommender system based on popularity → need for normalization.

Normalize co-occurrences: similarity matrix

- very similar to tf-idf

Jaccard similarity: normalizes by popularity
- Who purchased i and j divided by who purchased i or j



- Many other similarity metrics possible, e.g. cosine similarity

Limitations

- Only current page matters, no history

(27) Machine learning foundations - Washington

week 5

- Recommend similar items to the one you bought
- What if you purchased many items?
  - We want a new approach that includes the purchase history

(Weighted) Average of purchased items

- User bought items {diapers, milk} (only) ← purchase history
- Compute user-specific score for each item j in inventory by combining similarities:

$$\text{Score}(\text{user}, \text{baby wipes}) = \frac{1}{2}\left(S_{\text{baby wipes, diapers}} + S_{\text{baby wipes, milk}}\right)$$

(some approach but with milk raw)

look at the raw diapers and see how many times people also bought baby wipes

- Average the results and decide how likely the user will buy baby diaper given his purchase history.
- We could also weight recent purchases more and so on.
- Sort Score(user, j) and find item j with highest similarity

Limitations:

- Does not utilize: (well not directly)
  - context (e.g. time of day)
  - user features (e.g. age)
  - product features (e.g. baby vs electronics)

because looking at all users

(28) Machine learning foundation - Washington week 5

- **Cold start problem**
  - What if a new user or product arrives? How do we form recommendation if we have absolutely no information on that product / user?

Discovering hidden structure by matrix factorization (solution 3, recommender system)

Combines user history and product features and solves problems of new user / product.

Movie recommendation (content example)

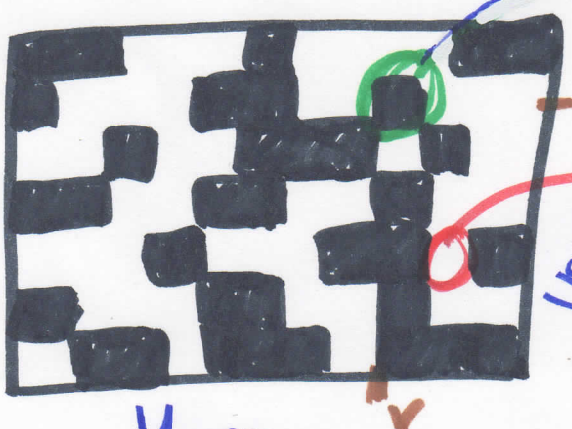
- Users watch movies and rate them

User	Movie	Rating
User 1	movie 1	★★★★
User 2	movie 2	★★★
User 3	movie 3	★★
User 4	movie 4	★
User 5	movie 5	★★★

- Each user only has watched a few of the available movies

Matrix completion problem (movie reco)

Rating =



unknown ratings

(29) Machine learning - Fundamentals - Washington weeks

- **Data**: Users score some movies
  - Rating (u, v) known for black cells
  - Rating (u, v) unknown for white cells
- **Goal**: Filling missing data (?) (white cells)
  - We use the known data (black cells) to fill in missing data (white cells)
  - Predict the unknown ratings here

Suppose we had d feature for each user and product (movie example) (=features)

- Describe movie v with topics R_v
  - How much is it action, romance, drama, ...

$R_v = [0.3 \quad 0.01 \quad 1.5 \ldots]$

- Describe user u with topics L_u
  - How much she likes action, romance, drama, ...

$L_u = [2.5 \quad 0 \quad 0.8 \ldots]$

Rating (u, v) is the product of the two vectors

estimate $R_v = [0.3 \quad 0.01 \quad 1.5 \ldots]$

$L_u = [2.5 \quad 0 \quad 0.8 \ldots]$

$L_{u'} = [0 \quad 3.5 \quad 0.01 \ldots]$

$\rightarrow 0.3 \times 2.5 + 0 + 1.5 \times 0.8 + \ldots = 7.2$

$\rightarrow 0 + 0.01 \times 3.5 + 1.5 \times 0.01 \ldots = 0.8$

(30) Machine learning - Foundations - Week 5

Recommendations: sort movies user hasn't watched by $\widehat{Rating}(u, v)$ — estimate

When the movie vector and the user vector agree a lot, we will get a larger number than when they don't.

higher rating — lower rating

We will sort all movies with their predicted ratings and recommend the ones with higher ratings.

With this type of model we are not restricted to keep the score (Rating estimate) between 0 and 5 (stars for movies), we just look the ones with higher scores

Predictions in matrix form

Result of

$\widehat{Rating}(u, v) = \langle L_u, R_v \rangle$

element-wise product

$Rating =$ [matrix: Users × Movies]

L_u (raw) = [. . . . . action Romance . . .]

L [matrix: Users × action Romance . . .]

R [matrix: Movies × action Romance . . .]

R_v = [. . . . . action Romance] (column)

(31) Machine learning_Foundations - Washington week 1

- But we don't know topics of users and movies..

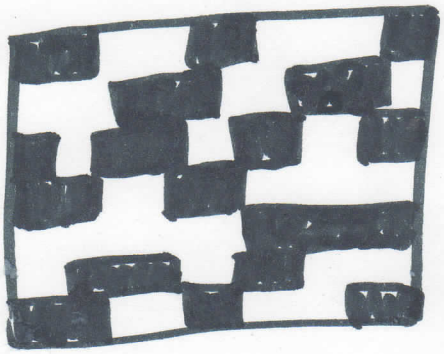
Matrix factorization model: discovering features from data

topics here

We are going to estimate L and R matrix (topics factors for every user and movie) based on an observed ratings. (=data)

L and R represents the parameters (similar to regression model that estimates parameters from data (black cells in the ratings matrix)

Rating =



≃